CLAIMS

What is claimed is:

	1	1.	A scheduling method comprising the steps of:
	2		based on scheduling states, defining a set of static schedules for an
	3		application;
	4		during run time, learning a cost of a set of static schedules based on
	5		performance of the application; and
	6		designating the static schedule with the lowest cost as an optimal
	7		schedule for the scheduling state.
նաք Կույք	1	2.	A scheduling method as claimed in Claim 1 wherein the cost of a set of static
B 4511	2		schedules is learned each time there is a change in scheduling state.
and and they the this and their their	1	3.	A scheduling method as claimed in Claim 1 wherein the cost of a set of static
	2		schedules is learned continuously during run time.
կուն հեռի հում։ Ռո .ուսի կուն	1	4.	A scheduling method as claimed in Claim 1 further comprising the steps of:
는 작년	2	٦.	storing a set of all possible schedules associated with each schedule state
	3		and
=	4		upon a change of state, selecting the optimal schedule associated with the
	5		schedule state.
	1	5.	A scheduling method as claimed in Claim 4 wherein the selected schedule is the
	2		schedule with the lowest cost.
	1	6.	A scheduling method as claimed in Claim 4 wherein the selected schedule is the
	2	0.	schedule with an unknown cost.
	-		

A scheduling method as claimed in Claim 6 wherein the schedule is randomly 7. 1 selected dependent on utility of exploration associated with the schedule. 2 A scheduling method as claimed in Claim 1 wherein the cost of a schedule is 1 8. computed and stored after the schedule is executed. 2 A scheduling method as claimed in Claim 1 further comprising the step of: 9. 1 maintaining a task execution cost for each task in the application for each 2 3 scheduling state. A scheduling method as claimed in Claim 9 wherein an optimal static schedule 1 10. associated with a new scheduling state is computed using stored task execution 2 3 costs. A scheduling method as claimed in Claim 10 wherein the cost of an individual 1 11. task is updated using a sliding window which discounts older execution results 2 3 at the expense of more recent execution results. A scheduling method as claimed in Claim 10 wherein the cost of a schedule is 12. 1 updated using a sliding window which discounts older execution results at the 2 expense of more recent execution results. 3 A scheduling method as claimed in Claim 1 further comprising the step of: 1 13. predicting the cost of a schedule dependent on stored task execution 2 3 costs. A scheduling method as claimed in Claim 13 wherein a schedule is selected for 14. 1 further exploration dependent on the predicted schedule cost. 2

	1	15.	A scheduling method as claimed in Claim 1 wherein the step of learning further
	2		comprises the steps of:
	3		storing application input data received during an active period in the
	4		application; and
	5		exploring optimal schedules while replaying the stored input data during
	6		an idle period in the application.
	1	16.	A scheduling method as claimed in Claim 15 wherein the step of learning further
	2		comprises the step of:
	3		concurrently executing a copy of an application with identical input data
**:	4		on a processor other than the processor on which the application is
and and the II I then mad had had	5		executing.
=	1	17.	A scheduling method as claimed in Claim 16 wherein a change in the optimized
Ų N	2		schedules is immediately reflected to the schedule analyzer for use in the next
ħ	3		schedule change of the application.
to the first of the second state			
i i	1	18.	A scheduling system comprising:
Ī	2		a set of static schedules for an application, the static schedules based on
	3		scheduling states; and
	4		a schedule analyzer which:
	5		during run time, learns a cost of the set of static schedules based
	6		on performance of the application; and
	7		designates the static schedule with the lowest cost as an optimal
	8		schedule for the scheduling state.
	1	19.	A scheduling system as claimed in Claim 18 wherein the schedule analyzer
	2	17.	learns the cost of a set of static schedules each time there is a change in
	3		scheduling state.

A scheduling system as claimed in Claim 18 wherein the schedule analyzer 1 20. learns the cost of a set of static schedules continuously during run time. 2 A scheduling system as claimed in Claim 18 further comprising: 1 21. a list of schedule costs which stores an optimal schedule associated with 2 each schedule state wherein upon a change of state the schedule analyzer selects 3 the optimal schedule corresponding to the schedule state. 4 A scheduling system as claimed in Claim 21 wherein the schedule analyzer 1 22. selects a schedule with the lowest cost. 2 A scheduling system as claimed in Claim 21 wherein the schedule analyzer 1 23. 2 selects a schedule with an unknown cost. A scheduling system as claimed in Claim 23 wherein the schedule analyzer 24. 1 randomly selects a schedule dependent on utility of exploration associated with 2 3 the schedule. A scheduling system as claimed in Claim 18 wherein the schedule analyzer 25. 1 computes the cost of a schedule and stores the computed cost after the schedule 2 3 is executed. A scheduling system as claimed in Claim 18 further comprises: 26. 1 a task execution table which stores a task execution cost for each task in 2 3 the application for each scheduling state. A scheduling system as claimed in Claim 26 wherein the schedule analyzer 1 27. computes an optimal static schedule associated with a new scheduling state 2 using stored task execution costs. 3

A scheduling system as claimed in Claim 27 wherein the schedule analyzer 1 28. updates the cost of an individual task using a sliding window by discounting 2 older execution results at the expense of more recent execution results. 3 A scheduling system as claimed in Claim 27 wherein the schedule analyzer 1 29. updates the cost of a schedule using a sliding window by discounting older 2 execution results at the expense of more recent execution results. 3 A scheduling system as claimed in Claim 18 wherein the schedule analyzer 30. 1 predicts the cost of a schedule dependent on stored task execution costs. 2 A scheduling system as claimed in Claim 30 wherein the scheduler analyzer 1 31. selects a schedule for further exploration dependent on a predicted schedule cost. 2 A scheduling system as claimed in Claim 18 further comprising: 32. 1 memory which stores application input data received during an active 2 period in the application, the stored application input data allowing the schedule 3 analyzer to explore optimal schedules while replaying the application input data 4 during an idle period in the application. 5 A scheduling system as claimed in Claim 32 wherein the schedule analyzer 1 33. provides a copy of an application and the stored application input data for 2 concurrent execution on a processor other than the processor on which the 3 4 application is executing. A scheduling system as claimed in Claim 33 wherein a change in the optimized 1 34. schedules is immediately reflected to the schedule analyzer for use in the next 2 3 schedule change of the application.

	1	<i>3</i> 3.	A scheduling system comprising.
	2		a set of static schedules for an application, the static schedules based on
	3		scheduling states;
	4		means for learning which during run time, learns a cost of a set of static
	5		schedules based on performance of the application; and
	6		means for selecting which designates the static schedule with the lowest
	7		cost as an optimal schedule for the scheduling state.
	1	36.	A scheduling system as claimed in Claim 35 wherein the means for learning
	2		learns the cost of a set of static schedules is learned each time there is a change
125 125 125	3		in scheduling state.
and and that I have seen him bear	1	37.	A scheduling system as claimed in Claim 35 wherein the means for learning
CA WE	2		learns the cost of a set of static schedules continuously during run time.
Hans Hans	1	38.	A scheduling system as claimed in Claim 35 further comprising:
:= i=	2		a list of schedule costs which stores an optimal schedule associated with
M	3		each schedule state wherein upon a change of state the means for analyzing
	4		selects the optimal schedule associated with the schedule state.
itat ²	1	39.	A scheduling system as claimed in Claim 38 wherein the means for selecting
	2		selects a schedule with the lowest cost.
	1	40.	A scheduling system as claimed in Claim 38 wherein the means for selecting
	2		selects a schedule with an unknown cost.
	1	41.	A scheduling system as claimed in Claim 40 wherein the means for selecting
	2		randomly selects a schedule dependent on utility of exploration associated with
	3		the schedule.

A scheduling system as claimed in Claim 35 wherein the means for selecting 1 42. computes the cost of a schedule and stores the computed cost after the schedule 2 is executed. 3 A scheduling system as claimed in Claim 35 further comprises: 1 43. a task execution table which stores a task execution cost for each task in 2 the application for each scheduling state. 3 A scheduling system as claimed in Claim 43 wherein the means for selecting 1 44. computes an optimal static schedule associated with a new scheduling state is 2 using stored task execution costs. 3 A scheduling system as claimed in Claim 44 wherein the means for selecting 1 45. updates the cost of an individual task using a sliding window by discounting 2 older execution results at the expense of more recent execution results. 3 A scheduling system as claimed in Claim 44 wherein the means for selecting 1 46. updates the cost of a schedule using a sliding window by discounting older 2 3 execution results at the expense of more recent execution results. A scheduling system as claimed in Claim 35 wherein the means for selecting 47. 1 predicts the cost of a schedule dependent on stored task execution costs. 2 A scheduling system as claimed in Claim 47 wherein the means for selecting 1 48. selects a schedule for further exploration dependent on the predicted cost for the 2 3 schedule. A scheduling system as claimed in Claim 35 wherein the on-line scheduling 1 49. 2 system further comprises:

	3		memory which stores application input data received during an active
	4		period in the application, the stored application input data allowing the
	5		scheduling analyzer to explore optimal schedules while replaying the application
	6		input data during an idle period in the application.
	1	50.	A scheduling system as claimed in Claim 49 wherein the on-line scheduling
	2		system provides a copy of a copy of an application and the stored application
	3		input data for concurrent execution on a processor other than the processor on
			which the application is executing.
	1	51.	A scheduling system as claimed in Claim 18 wherein a change in the optimized
1	2		schedules is immediately reflected to the means for analyzing for use in the next
and and that I is then and that that	3		schedule change of the application.
	1	52.	A computer system comprising:
Ŧ	2		a central processing unit connected to a memory system by a system bus;
<u>-</u>	3		an I/O system, connected to the system bus by a bus interface; and
ման ման նում եր Մուսան հույն	4		a scheduling system routine located in the memory system which:
# #	5		based on scheduling states, defines a set of static schedules for an
	6		application;
=	7		during run time, learns a cost of a set of static schedules based on
	8		performance of the application; and
	9		designates the static schedule with the lowest cost as an optimal
	10		schedule for the scheduling state.
	1	53.	A computer program product for system scheduling, the computer program
	2	JJ.	product comprising a computer usable medium having computer readable
	3		program code thereon, including program code which:
	-		F0

The same of the sa	
j	
÷ 24.5	
::::::::::::::::::::::::::::::::::::::	=
The state of	
Ē	
100	
N III	
1	
101	
in:	
10:	-

4	based on scheduling states, defines a set of static schedules for ar
5	application;
6	during run time, learns a cost of a set of static schedules based on
7	performance of the application; and
8	designates the static schedule with the lowest cost as an optimal
9	schedule for the scheduling state.